

HL7 with CORBA and OLE: Software Components for Healthcare

Wes Rishel

Co-Chair, HL7 Special Interest Group for Object Brokering Technologies.

Componentized software promises easier, more fine-grained integration of disparate software systems. Variations of the technology can help to achieve tight coupling among disparate programs on the clinical workstation or across wide area networks. HL7 members have been designing extensions to the protocol for the exchange of healthcare information using Microsoft OLE and CORBA technologies. Extensive prototyping has been performed, including the simultaneous interconnection of sixteen different vendor systems exchanging demographic data and lab results. The first release of this standard will be notable in that the specifications for OLE and CORBA will be entirely isomorphic, they will be based directly on HL7 version 2.3, and they may easily be implemented in systems that are not written using object-oriented programming tools. As HL7 version 3 is developed on an object-oriented model of healthcare information, the same approach will be used so information about the objects may be shared using CORBA and OLE.

Background

The information systems industry is undergoing one of its frequent paradigm shifts. The new paradigm is software components.¹ Depending on who you ask the old paradigm was “cut and paste” (primitive data sharing), client-server (use of server technologies to share data, with little ability to share knowledge or business rules), or the monolithic application that may do everything but doesn’t to it all well. (“Buy ours because it has more features on the check-list than Brand X.”)

Software component interfaces are used today for many prosaic tasks such as dragging a file name into a mail client to attach a document, including video clips or high-quality audio in presentations, or creating a spreadsheet that automatically updates as new data arrives from a market price source.

A more futuristic scenario might be a special purpose application that is built by gluing together pieces of a word processing application, a specialized drawing tool, third-party spell-checkers, and repositories of boiler-plate text and art work, (e.g., WordPerfect for Steam Turbine Inspectors).

Another future scenario might include a “knowledge component” that is distributed nationally with con-

tinuously updated information on drug-drug interactions. All of these components could be used in many diverse applications even though the applications were written in different languages using different technologies.

These scenarios have some common threads. They combine pieces of different developers’ software into a smoothly functioning application. They also combine locally written work (the report, the spreadsheet, or perhaps the healthcare application) with vended software or data resources.

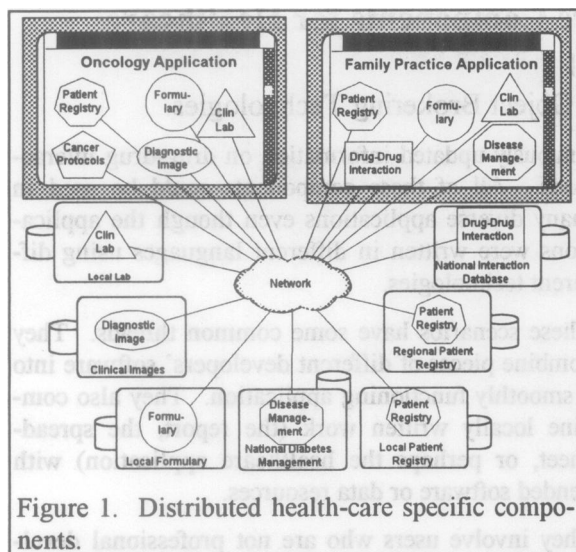
They involve users who are not professional developers being able to accomplish substantial tasks. This is because the high-power programming required to deal with a network, real-time data sources, multimedia data, or access an on-line national database is encapsulated into components. Through componentry, the less experienced programmers and power users can access these functions.

“The Grand Canyon”

In this section we paint a picture of the upside potential of this paradigm shift. We invite the user to examine it uncritically, to get an appreciation of how the benefits might justify the investment. In a later section we will describe more short-term efforts.

Figure 1 provides a view of the potential. We think of it as our Grand Canyon view, positively awesome. Oncologists have built an oncology system, family practitioners are using a system built by one of their own, and yet they are all sharing the common corporate data and functions. Patient Registry, for example, is more than a database. It implements a set of rules to maintain data integrity. The rules may be different for the regional patient registry, but this is not a problem because they are implemented by the regional component. Similarly, the diagnostic image component includes the special programming it takes to manipulate images and render them with clinically acceptable resolution. Knowledge components are integrated with national sources of disease management and drug-drug interactions. This aggregation of specialized components is integrated into unified user interfaces by the specialized applications.

The full picture would contain nursing systems, systems for case managers, and executives each with its



specialized mix of unique and shared data and functionality.

For some, the awesome vision of the ultimate health-care application may be different. Perhaps it would have a single user interface that is used by all caregivers. This should not be a problem. Since the group that defines that program need not reinvent basic notions of patient registry, disease management, image manipulation, etc., the cost of entry is relatively low. The industry should support developers with a variety of philosophies.

The Grand Canyon wasn't dug in a day. There'll be a lot of water down the river, a lot of technologies matured, and a lot of standards developed before this vision is real. And yet, less sweeping variations on this theme are in development today. Developers are building new systems using componentized technologies, and seeking to expose more fine-grained objects than the whole system. They are seeking tightly coupled, componentized interfaces among their own products and their collaborators. At the same time they are designing more loosely coupled interfaces for legacy systems.

Local Scenery, the Clinical Workstation

Sometimes we may get more enjoyment from our local scenery than the Grand Canyon, because it's easier to get there. We can realize the benefits of componentized software sooner if our goals are less sweeping. The Clinical Workstation could be such a venue. Although few would agree on its exact definition, most would agree with the following statements.

It's a multi-application platform based on a personal computer. The workstation runs more than a

single application simultaneously. There may be several applications that share the workstation, and the users will use the standard productivity tools available to PC users.

It should be easy to share data and context among the workstation applications. Users are getting used to cutting and pasting and dragging and dropping among dissimilar applications and will want similar levels of integration among their healthcare applications. The user should not have to log in separately to each application, and yet security must be maintained and enhanced. If a patient is selected in one application, other applications should not be showing data from another patient. They should blank or switch to the selected patient.

Others would agree with the above statements but want more. They see power users developing personal and departmental applications using productivity tools and software components on the workstation. Component technology means that word-processors, spreadsheets, and drawing tools no longer need be treated as monolithic entities.

Still others would find troublesome this notion of power users developing applications. They have looked for ways to keep this from happening, out of concerns for data integrity and reliance on data formats that makes it impossible to introduce change.

Software components may help alleviate their concern. Major complex applications like care path-based order management will not be written by "power users." But hundreds of requests for data extracts, for specialized data collection, for specialized management reports, etc., could be removed from the IS department queue and left to the resources of the departments that need them.

To this end, these same managers would add these requirements to the clinical workstation. *It should provide power users access to the corporate health-care data resource with proper security and without creating a nightmare of dependencies on the specific technologies and products used to build and maintain the data.* They see software components as a means to provide a middleware layer that isolates the user programs from data formats and database technologies.

When users' needs can be met by combining components from the general computing industry with healthcare components, some see the creation of a new market. Developers are freed from the need to field a full application and compete on the basis of a

“feature checklist.” They can create specialized endeavors bringing forth *knowledge components* to deal with drug, disease management, complex diagnosis, and many other highly specialized areas.

Component Technologies

There is a competition among two approaches to providing the componentized software technology. At one end of the playing field we have Microsoft, with its OLE. At the other we Object Operating Management Group (OMG) with CORBA.

The playing field has two zones, roughly corresponding to our Grand Canyon and Local Scenery sections, above. For the Local Scenery, intra-workstation componentry is critical including the ability to access software components from a wide variety of programming languages and productivity tools. For the Grand Canyon, facilities that can coordinate distributed processing are critical. These include wide area brokering services, and the derivative services necessary to support transaction processing and concurrent database updates across very wide area networks.²

Microsoft uses OLE³ for all manner of software component interfaces in Windows. It is widely used today for everything from controlling multimedia devices and telephones, to making artificial intelligence software accessible in a wide variety of different applications. Until recently it was limited to interfaces that are local to a workstation. Microsoft has begun to deliver the most basic extensions of OLE to provide interfaces over networks. It has announced further extensions to provide the derivative service needed for the Grand Canyon scenarios, and to bring a subset of OLE to other operating systems.

The OMG, a computer-industry consortium, is writing standards for componentized software over large networks and across operating systems. The rubric for this family of standards is the Common Object Request Broker Architecture (CORBA)⁴. It includes standards for the basic services necessary to share software objects and the derivative services for large-scale distributed processing. CORBA advocates have acknowledged that early specifications and software were neither interoperable nor sufficiently robust for large-scale networked applications.⁵ CORBA version 2 provides interoperability and is claimed to support large scale distributed applications. Software tool sets that implement version 2 are beginning to reach the marketplace in 1996. Initial implementations of the software to support the

derivative services necessary for wide area networks are expected soon.

The Health Level-7 Approach

The Health Level-7 Special Interest Group on Object Brokering Technology (SIGOBT) has been investigating these technologies for two years. The group has made substantial progress through prototyping and analysis. Figure 2 illustrates a class of component interfaces which HL7 can introduce into the Clinical Workstation. They would greatly facilitate some of the “Local Scenery” goals of the Clinical Workstation. Two qualitatively different kinds of components are shown. There are complete applications, with their own GUIs. There are also components that have no visible on-screen manifestation. These either serve as the proxy for a legacy system or provide special coordination services.

The applications and the invisible objects connect through two kinds of interfaces. The “data or function” interfaces let components share data or control the actions of each other actions. The context interface would let the applications use the Clinical Context Object to coordinate their context. The context interfaces support the context sharing features described in Figure 2 and the “data or function” interfaces support “healthcare data component” and “automation interface” features.

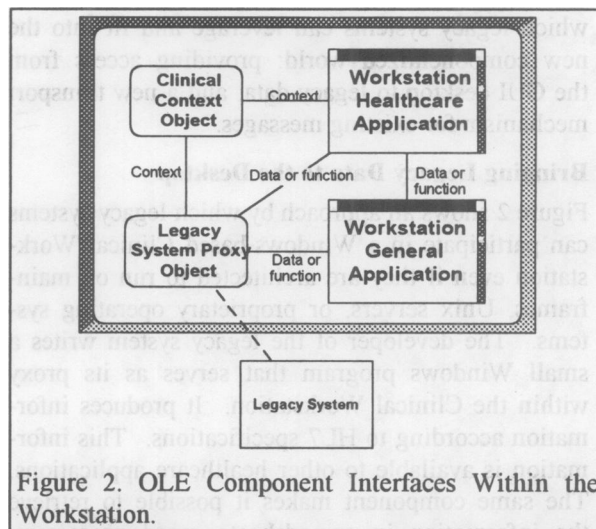


Figure 2. OLE Component Interfaces Within the Workstation.

Current HL7 SIGOBT work is limited to the data or function interfaces. The author has proposed the Clinical Context Object as a future agenda item for SIGOBT.

Figure 3 illustrates how software component technology can produce incremental improvements in

current HL7 applications. By using CORBA to send standard HL7 messages, sites would be relieved of many of the chores of establishing the communications format, mapping applications to one another and maintaining the links. The figure also illustrates a straightforward adaptation whereby healthcare application protocols built on a Windows operating system could use an OLE interface while interoperating with CORBA clients.

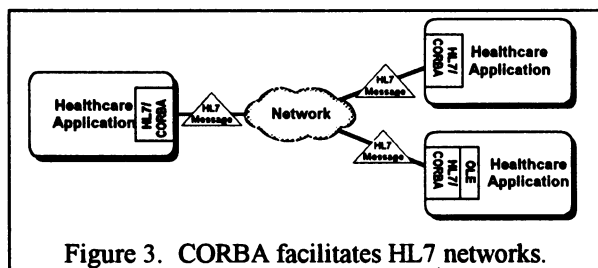


Figure 3. CORBA facilitates HL7 networks.

An important characteristic of the current HL7 work is that the specifications for the CORBA and OLE interfaces described in Figure 3 and the “data or function” interfaces described in Figure 2 are virtually identical. This provides substantial leverage against future events where networked OLE becomes an established means of communications or users want to use CORBA within the workstation.

Support for Legacy Systems

There are two ways currently being explored in which legacy systems can leverage and fit into the new componentized world: providing access from the GUI desktop to legacy data, and a new transport mechanism for existing messages.

Bringing Legacy Data to the Desktop

Figure 2 shows an approach by which legacy systems can participate in a Windows-based Clinical Workstation even if they are architected to run on mainframes, Unix servers, or proprietary operating systems. The developer of the legacy system writes a small Windows program that serves as its proxy within the Clinical Workstation. It produces information according to HL7 specifications. This information is available to other healthcare applications. The same component makes it possible to retrieve the information in spreadsheets, word processors, and using Visual Basic, Delphi, Visual C++, and other programming environments that support OLE components. This approach makes possible the “data or function sharing” interfaces.

Object Request Brokers Simplify Messaging

In current HL7 implementations each developer must create software to compose and parse messages. This adds to the cost of interfaces and creates the potential errors. A benefit can be achieved by using an Object Request Broker to communicate HL7 messages without the need to encode them.

Figure 3 is an example of such an architecture in which HL7 messages are sent between systems using CORBA. The Andover Group for Open Healthcare Interoperability is an industry consortium which is developing specifications to supplement HL7 and software to implement this architecture.

Figure 4 is an adaptation of Figure 3, showing a strategy for adding CORBA-based message delivery to an environment that is already using a gateway to route the transactions. This kind of mapping is possible because the SIGOBT architecture for HL7 version 2.2 will be based directly on the existing message structures.

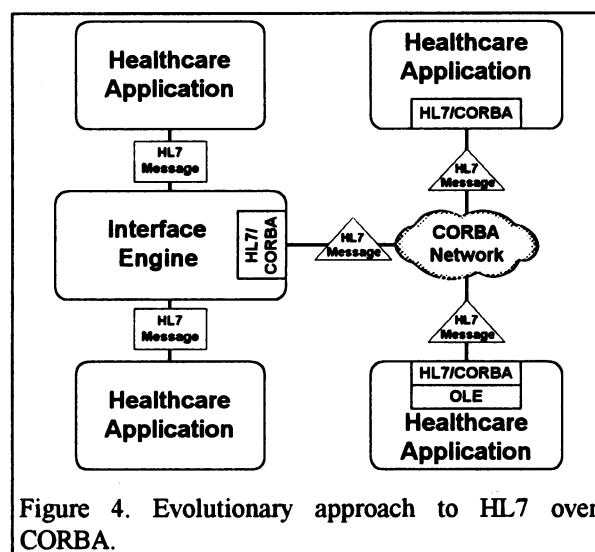


Figure 4. Evolutionary approach to HL7 over CORBA.

The Design Model

The data or function sharing interfaces of Figure 2 have been explored intensively in the prototyping work of the HL7 SIGOBT group. This approach is illustrated in Figure 5. Solid lines represent OLE interfaces. Broken lines indicate the manufacture of objects. Healthcare applications offer to be *producers* of healthcare information, the *consumers* or both. A producer provides healthcare data. So a lab application might offer to share results about patients through the HL7 object. It would respond to queries from consumers by returning those results. A clinical system might be a consumer in order to integrate

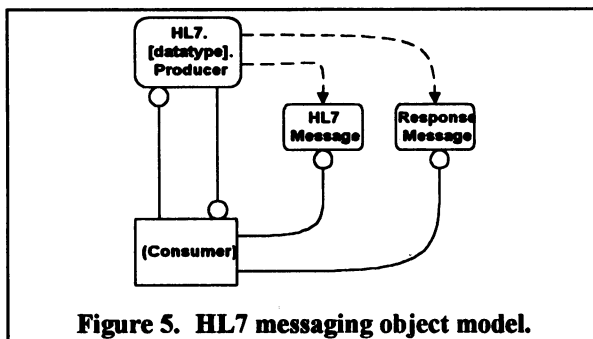


Figure 5. HL7 messaging object model.

those results into its patient information displays. At the same time it might be a producer, offering, perhaps, vital signs and other observations collected through its user interface to other applications.

The methodology that has been employed is based on shared objects that directly relate to messages in HL7 version 2.2 (and ultimately, 2.3). As shown in Figure 5, there is a component which represents the producer of a specific kind of data (e.g., laboratory data.) This is a first class object. It is known to the object broker and can be invoked by another program even though it is a separate component whose specific location is unknown. The HL7 Consumer is any program that invokes the object. The Producer creates second class objects that are directly isomorphic to HL7 messages. The Consumer accesses the message object to populate its fields and tells the message to send itself. The Producer creates a response message object which can be examined by the consumer to recover the information. The architecture also supports unsolicited transmission of information from the Producer to the Consumer.

Prototyping Activities and Next Steps

This architecture was described in a concept paper. It was developed into the initial Object Mapping Specification in a prototype specification based on version 2.2 and OLE. Microsoft contributed sample code.^{*} Sixteen healthcare developers adapted their applications to exchange lab results using this specification. Interoperability testing among all sixteen developers was completed successfully in a single day.

At the completion of the prototype the SIGOBT group began work using the same methods to develop a robust specification for using components to

exchange a subset of all the kinds of data that are covered by HL7 version 2.3 specifications.

One of the main differences between HL7 version 2.3 and version 3, is that the latter standard will be derived from an object-oriented information model. HL7 is currently developing the *HL7 Reference Model* and working to harmonize it with the models of other healthcare standards groups through the Joint Working Group for a Common Data Model⁶. When the information model is complete it will be communicated using the same design model.

Summary

HL7 is developing an adaptation of its standard to operate over CORBA and OLE object brokering technologies. The approach is notable in that it applies equally to the integration within the Clinical Workstation and across very wide area network, using a common object model for both milieus and brokering technologies. Prototyping activities based on OLE have been successful with very short integration testing times. This work will be used in production standards based on HL7 version 2.3 (which is not based on an object oriented information model) and version 3 (which is).

References

- ¹Udell J Componentware. Byte 1994;19(5):52.
- ²Orfali R, Harkley D, Edwards J The Essential Distributed Object Survival Guide. New York: John Wiley, 1996:123-202.
- ³Brockschmidt K Inside OLE—Second Edition. Microsoft Press 1995:23-51.
- ⁴Operating Management Group. CORBA: Architecture and Specifications. 1995.
- ⁵Orfali: 62.
- ⁶IEEE P1157.1 Joint Working Group for a Common Data Model. Trial-Use Standard for Health Care Data Interchange — Information Model Methods, pre-ballot draft dated April 1996.

^{*} Various white papers, specifications and sample code are available from the HL7 Web Server. <http://dumccss.mc.duke.edu/ftp/standards.html>